

Massively Parallel Simulation of Flow and Transport in Variably Saturated Porous and Fractured Media

Yu-Shu Wu, Keni Zhang, and Karsten Pruess

Earth Sciences Division, Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA

Abstract

This paper describes a massively parallel simulation method and its application for modeling multiphase flow and multicomponent transport in porous and fractured reservoirs. The parallel-computing method has been implemented into the TOUGH2 code and its numerical performance is tested on a Cray T3E-900 and IBM SP. The efficiency and robustness of the parallel-computing algorithm are demonstrated by completing two simulations with more than one million gridblocks, using site-specific data obtained from a site-characterization study. The first application involves the development of a three-dimensional numerical model for flow in the unsaturated zone of Yucca Mountain, Nevada. The second application is the study of tracer/radionuclide transport through fracture-matrix rocks for the same site. The parallel-computing technique enhances modeling capabilities by achieving several-orders-of-magnitude speedup for large-scale and high resolution modeling studies. The resulting modeling results provide many new insights into flow and transport processes that could not be obtained from simulations using the single-CPU simulator.

1. INTRODUCTION

In recent years, the demand on modeling capability in the fields of oil, gas, and geothermal reservoirs, groundwater resource management, and subsurface contamination investigation has increased rapidly, as required by efforts in detailed site characterization and reliable model prediction. Limitations of traditional single-CPU simulators have prompted increased attention to massively parallel techniques for reservoir simulation and groundwater modeling communities [1]. Motivated by the desire for large-scale, realistic reservoir simulations, research on parallel reservoir simulation started in the early 1980s. Those earlier efforts [2-3] were focused on improving modeling tools using the vectorization of supercomputers to enhance modeling capabilities. From the late 1980s to the early 1990s, significant progress was made in improvement of parallelization algorithms [4], implementation into simulators using different computers [5], and applications to large-scale reservoir simulations [6].

In the early 1990s, more sophisticated and efficient parallel reservoir simulators were developed [7]. By the late 1990s, the parallel-computing reservoir simulation technology was further improved [8]. Realistic field applications of parallel techniques were demonstrated with multimillion gridblock reservoir simulations [9]. More recent developments in parallel

simulations include high-performance simulation of groundwater flow and contaminant transport [10], multiphase flow [11,12], and new algorithm development [13].

This paper presents our continuing effort to improve parallel schemes for large-scale multiphase reservoir simulation. The new development discussed here involves adding the capability of handling tracer or radionuclide transport on to a parallel scheme [11,12]. The implemented scheme is a machine-independent algorithm, which can be easily ported into a mainframe supercomputer, a multiprocessor workstation, or a cluster of PCs or workstations. In particular, the proposed parallel scheme is implemented into the TOUGH2 family of codes [14] and tested on Cray T3E and IBM SP massively parallel-processing (MPP) computers.

Application of the parallel-computing scheme is demonstrated through modeling of two unsaturated flow and transport problems at the Yucca Mountain site, a potential repository site for a high-level nuclear waste for the U.S. In addition, these test problems are also used to evaluate the numerical performance of the parallel-computing scheme. Simulation results for the two applications presented below indicate that the parallel-computing technique implemented in the TOUGH2 code is very efficient in both computing speedup and memory usage.

2. METHODOLOGY

The massively parallel-computing technique of this work is based on a fully implicit formulation of the single-CPU version of the TOUGH2 code [14]. This is because the fully implicit scheme has proven to be the most robust numerical approach in modeling highly nonlinear multiphase flow and heat transfer in reservoir simulations. For a typical fully implicit simulation using Newton iteration, the most time-consuming parts of the execution consist generally of two parts: (1) assembling the Jacobian matrix and (2) solving the resulting linearized system of equations. Consequently, one of the most important aims of a parallel code is to distribute computational efforts for these two tasks. In addition, a parallel scheme should also take into account grid node/element domain partitioning, grid node/element reordering, data input and output optimizing, and efficient message exchanging between processors.

The first important step in a parallel simulation is to balance computational efforts among the processors by distributing gridblocks evenly. Secondly, the time consumed in communication between processors should be minimized when solving a globally coupled equation system. In the TOUGH2 simulation, a model domain/grid system is represented by a set of three-dimensional gridblocks and their connections. The entire system of gridblocks is treated as an unstructured grid. We use one of the three partitioning algorithms (*K-way*, *VK-way*, and *Recursive*) of the METIS package (version 4.0) [15] for our grid domain partitioning [11,12]. In addition to parallelization of computation and communication efforts, one should also consider how to handle memory requirements for storing input data, which, including hydrogeologic parameters and initial and boundary conditions, could become too large for a large-scale, three-dimensional, heterogeneous reservoir to be handled by one processor. Therefore, the memory load also needs to be distributed to all processors.

The fundamental goal of a reservoir simulator is to solve spatially and temporally discretized, nonlinear governing equations for flow and transport processes in porous media. These discrete mass- and energy-balance equations can in general be written in residual form [14]:

$$R_n^\kappa(x^{t+1}) = M_n^\kappa(x^{t+1}) - M_n^\kappa(x^t) - \frac{\Delta t}{V_n} \left\{ \sum_m A_{nm} F_{nm}^\kappa(x^{t+1}) + V_n q_n^{\kappa,t+1} \right\} = 0 \quad (1)$$

where the vector x^t consists of primary variables at time t , R_n^κ is the residual of component κ (heat is also regarded as a “component”) for block n , M denotes mass or thermal energy per unit volume for a component, V_n is the volume of the block n , q denotes sinks and sources of mass or energy, the current time step size is denoted by Δt , $t+1$ denotes the current time, A_{nm} is the interface area between blocks n and m , and F_{nm} is the “flow term” between them (Note that the flow term includes both advective and diffusive [conductive for heat] fluxes. Equation (1) is solved using the Newton method, leading to

$$-\sum_i \frac{\partial R_n^{\kappa,t+1}}{\partial x_i} \bigg|_p (x_{i,p+1} - x_{i,p}) = R_n^{\kappa,t+1}(x_{i,p}) \quad (2)$$

where $x_{i,p}$ represents the value of the i th primary variable at the p^{th} iteration.

The Jacobian matrix, as well as the right-hand side of (2), needs to be updated at each Newton iteration, and thus computational efforts may be extensive for a large simulation. In the parallel scheme, the assembly of the linear equation system (2) is shared by all the processors. The resulting linearized equation system arising at each Newton step is then solved using an iterative parallel linear solver from the Aztec package [16]. Note that dynamic memory management, modules, array operations, matrix manipulation, and other FORTRAN 90 features are implemented in the parallel code. In particular, the message-passing interface (MPI) library [17] is used for message passing.

In the parallel code, initial and boundary conditions are handled by taking advantage of the TOUGH2 methodology and are carried out during system initialization and/or grid partitioning. For example, first-type or Dirichlet boundary conditions are treated using the large-volume method, in which the boundary node is specified with a numerically large volume. Once specified, primary variables will be fixed at the large-volume boundary nodes, and the code handles these boundary nodes exactly like any other computational nodes. This treatment is easily implemented by the parallel-computing scheme through grid-domain partitioning.

The TOUGH2 formulation and parallel implementation are applicable to both single-continuum and multicontinuum media. TOUGH2 handles fractured reservoirs by preprocessing a mesh to represent fracture and matrix domains separately. This is directly applied to a parallel situation. Once a proper mesh for the fracture-matrix system is generated, fracture and matrix blocks are specified to represent fracture or matrix domains, respectively. Formally, they are treated on the same footing during grid partitioning or solution in the model (i.e., they are handled as a special case of the unstructured grid).

3. APPLICATION

We present two examples to investigate computational performance and to demonstrate application of the proposed parallel simulation method.

3.1. 3-D Flow Simulation

The problem is based on a large-scale 3-D flow model for modeling flow within the unsaturated zone at Yucca Mountain, Nevada [18]. Here the problem is studied using a much-refined grid to evaluate the numerical performance of the parallel scheme and to demonstrate its application. The problem [18] concerns unsaturated flow through fractured rock under ambient conditions, using a 3-D, unstructured grid and a dual-permeability approach for handling fracture-matrix interactions. The model domain and the numerical grid, encompassing approximately 40 km² of the Yucca Mountain area, as shown in plan view in Figure 1. There are approximately 9,900 blocks per grid layer and about 60 computational grid layers vertically from land surface to water table. This results in a total of 1,100,000 gridblocks for fractures and matrix, and 4,050,000 connections. A distributed-memory Cray T3E-900 computer equipped with 695 processors was used for this simulation example.

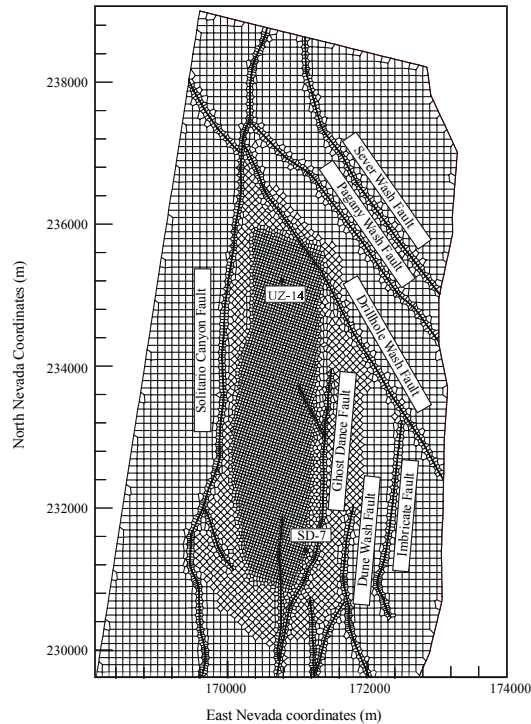


Figure 2. Plan view of the 3-D site-scale model domain, grid and incorporated major faults

In the example, the ground surface, the top boundary, is specified with spatially varying water infiltration, with an average infiltration rate of 4.6 mm/yr over the model domain. The bottom boundary is described as a water table, while the surrounding lateral boundaries are treated as closed boundaries. The properties used for fractures and rock matrix for the dual-permeability model, including two-phase flow parameters of fractures and matrix, were estimated from field tests and model calibrations [18].

Using 64 processors, the simulation was completed with 3,684 time steps to reach a steady-state solution. Figure 2 shows a comparison of the simulated flux distributions along the repository, using the refined-grid model (Figure 2a) and comparing the results to the coarse-grid results from 100,000 gridblocks for the same geological domain (Figure 2b [18], nonparallel version). Comparison of the simulation results in Figures 2a and 2b indicates that the two models predict very different flow patterns in the repository area (along the middle refined-grid portion of the model domain—see Figure 1). The refined-grid model predicts much smaller percolation fluxes within the repository area. These results could have a direct impact on performance assessment.

Numerical performance for simulating the same unsaturated flow problem using the refined grid model was evaluated using 32, 64, 128, 256, and 512 processors, respectively, for 200 time-steps. Table 1 shows the improvement (or reduction) in the total execution time as the processor number increases. The results clearly indicate that the execution time is significantly reduced with the increase in number of processors. Table 1 also lists the times used for different computational tasks using different numbers of processors. From the table, we can see that the best parallel performance is achieved in solving linear-equation systems. Note that data input and output of the program were carried out using a single processor; using a different number of processors might produce different overheads. In comparison, the time requirement for the model-setup phase (input, distribution, and initialization) in Table 1 increases when the processor number is doubled from 256 to 512 (instead of generally decreasing). It indicates that a saturation point has been reached, due to the increased communication overhead when increasing the number of processors. Nevertheless, the parallel scheme achieved two-orders-of-magnitude speedup when compared with the single-CPU code.

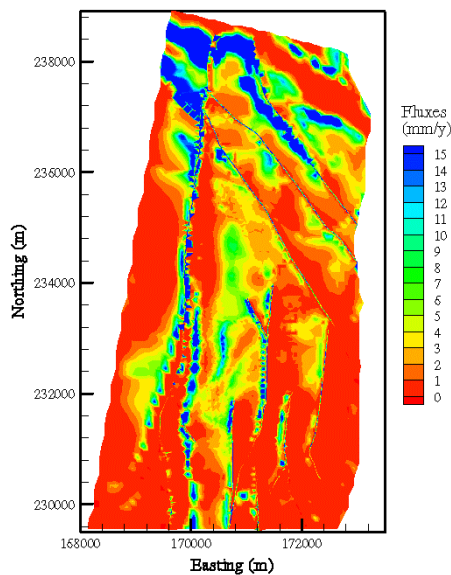


Figure 2a. Distribution of simulated vertical liquid fluxes at repository horizon: (a) using the refined-grid model (Figure 1)

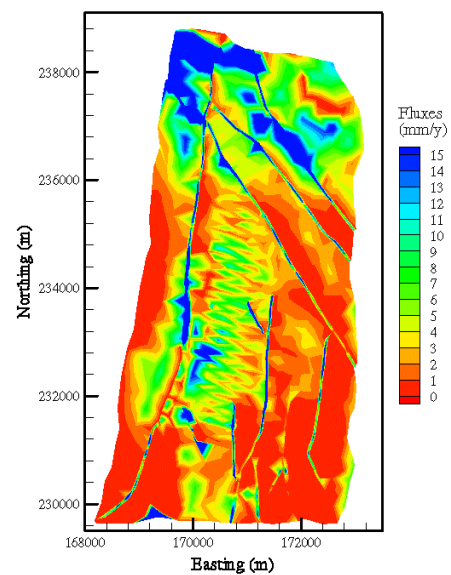


Figure 2b. Distribution of simulated vertical liquid fluxes at repository horizon: (b) using the coarse-grid model [18]

Table 1

Statistics of Execution Times and Iterations for the 3-D Site-Scale Flow Problem of for 200 Time Steps, Using Different Numbers of Processors

Number of processors	32	64	128	256	512
Input, distribution, and initialization (s)	592.3	248.1	116.5	84.3	134.3
Update thermophysical parameters and setup Jacobian matrix (s)	2659.2	1420.8	764.6	399.5	260.0
Solve linear equations (s)	6756.7	2078.7	806.6	373.4	188.0
Total execution time (s)	10100.5	3844.3	1780.8	950.6	618.0
Total Newton iterations	415	415	424	424	423
Total Aztec iterations/processor of solving linear equations	5059	5520	5735	6353	6281

3.2 3-D Transport Simulation

The second example demonstrates application and efficiency of the parallel code for modeling 3-D tracer/radionuclide transport within the unsaturated zone at Yucca Mountain, using the steady-state, 3-D flow field of Section 3.1. However, the calculation was run to 2,000,000 years using 64 processors only, and another distributed-memory IBM SP computer with 208 16-processor nodes (with a total of 3,328 processors) is used.

Tracer or radionuclide is treated as a conservative component. The mechanical dispersion effect through the fracture-matrix system is ignored, and a constant molecular diffusion coefficient of 3.2×10^{-11} (m²/s) is used for matrix diffusion. The transport simulation is run to 2,000,000 years with a constant initial concentration source, released at the repository from fracture or matrix blocks.

A cumulative mass breakthrough curve of the tracer at the water table (calculated from the simulation results), is shown in Figure 3. The cumulative mass breakthrough in the figure is defined as the cumulative mass of tracer or radionuclide arriving at the water table over the entire bottom model boundary over time, normalized by the total initial mass of the component at the repository. Figure 3 indicates that on average it takes thousands of years for the tracer to travel from the repository to the water table.

Table 2 qualifies parallel computing performance in simulating transport simulation of the initial radionuclide release from fractures. A comparison of the execution times, spent in different portions during the two-million year simulation shown in Table 2, indicates that solving the linear equations takes a much smaller percentage (17%) of the total simulation time than that for the flow simulation. This is because the transport problem becomes simply linear with computational options selected.

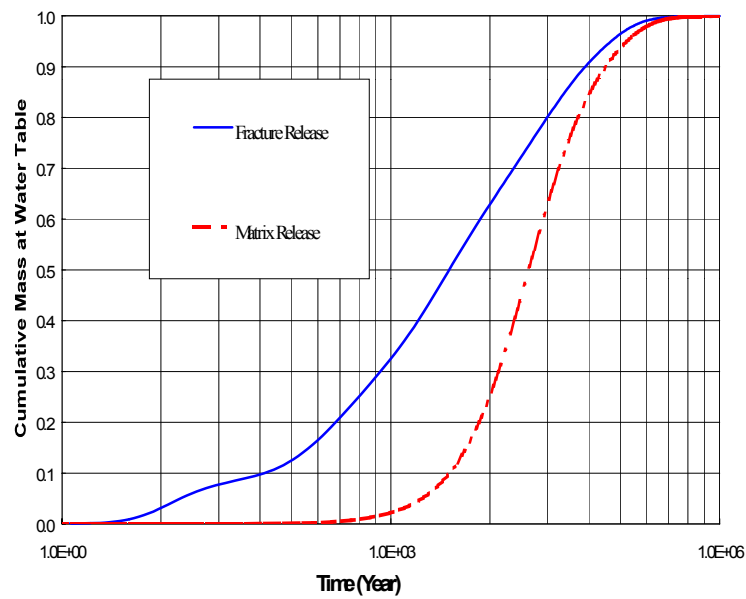


Figure 3. Simulated breakthrough curves of cumulative tracer/radionuclide mass arriving at the water table, after release from fracture and matrix blocks at the repository.

Table 2
Summary of Execution Times and Iterations for the Transport Simulation with Fracture Release Using 64 Processors

Input, distribution, and initialization (s)	139.7
Update thermophysical parameters, Jacobian matrix (s)	333.7
Solve linear equations (s)	124.5
Total execution time (s)	727.7
Total Newton iterations	118
Total time steps	118
Total Aztec iterations/PE of solving linear equations	1,096

4. SUMMARY

This paper describes a massively parallel simulation technology. The technique involves a machine-independent parallel-computing algorithm using a highly portable communication system, providing a framework applicable to a mainframe supercomputer, a multi-processor workstation, or a cluster of PCs or workstations. To demonstrate the proposed parallel-computing method, we have presented two large-scale applications of flow and transport in

the unsaturated zone at Yucca Mountain. Test results concluded that the proposed massively parallel-computing method greatly improved computational efficiency and robustness for dealing with highly nonlinear problems, such as large-scale reservoir simulations. Parallel-computing techniques can particularly enhance modeling capabilities, achieving several-orders-of-magnitude speedup for large-scale and high-resolution modeling studies.

ACKNOWLEDGEMENTS

The authors would like to thank Andre Unger and Dan Hawkes for their review of this paper. This work was supported by the Director, Office of Civilian Radioactive Waste Management, U.S. Department of Energy, through Memorandum Purchase Order EA9013MC5X between Bechtel SAIC Company, LLC and the Ernest Orlando Lawrence Berkeley National Laboratory (Berkeley Lab). The support is provided to Berkeley Lab through the U.S. Department of Energy Contract No. DE-AC03-76SF00098.

REFERENCES

1. P. Oduro, H.V. P, Nguyen and J.L. Nieber, Proceedings of the 1997 ASAE Annual International Meeting. Part 1 (of 3), Minneapolis MN, (1997) 0154.
2. A.J. Scott, B.R. Sutton, J. Dunn J, P.W. Minto, and C.L. Thomas, Proceedings of Sixth SPE Symposium on Reservoir Simulation, New Orleans, Louisiana, (1982) 523.
3. J.E. Killough and J.M. Levesque, Proceedings of Sixth SPE Symposium on Reservoir Simulation, New Orleans, Louisiana, (1982) 481.
4. J. Barua and R.N. Horne, Proceedings of Tenth SPE Symposium on Reservoir Simulation, Houston, Texas, (1989) 7.
5. J.A. Wheeler and R.A. Smith, SPE Reservoir Engineering, (1990) 544.
6. S.L. Scott, R.L. Wainwright, R. Raghavan, and H. Demuth, Proceedings of Ninth SPE Symposium on Reservoir Simulation, San Antonio, Texas, (1987) 1.
7. K. Hemanth-Kumar and L.C. Young, Proceedings of 13th SPE Symposium on Reservoir Simulation, San Antonio, Texas, (1995) 12.
8. M.F. Wheeler, T. Arbogast T, S. Bryant, J. Eaton. Q. Lu, M. Peszynska, and I. Yotov, Proceedings of Fifteenth SPE Symposium on Reservoir Simulation, Houston, Texas, (1999) 14.
9. A.H. Dogru, Journal of Petroleum Technology, **52**(5), (2000) 54.
10. H. Zhang, F.W. Scharitz, and E.A. Sudicky, Water Resources Research, **30**(12), (1994) 3553.
11. K. Zhang, Y.S. Wu, C. Ding, K. Pruess, and E. Elmroth, Proceedings of the 2001 SPE Reservoir Simulation Symposium, Houston, Texas, (2001).
12. Y.S. Wu, K. Zhang, C. Ding, K. Pruess, E. Elmroth, and G.S. Bodvarsson, (in press) Journal of Advances in Water Resources, (2002).
13. S.F. Ashby and R.D. Falgout, Nuclear Science and Engineering, **124** (1), (1996) 145.
14. K. Pruess, C. Oldenburg, G. Moridis, Lawrence Berkeley Laboratory Report LBNL-43134, Berkeley, CA, (1999).
15. G. Karypis and V. Kumar, Technical Report, Department of Computer Science, University of Minnesota, (1998).
16. R.S. Tuminaro, M. Heroux, S.A. Hutchinson, and J.N. Shadid, Sandia National Laboratories, Albuquerque, NM, (1999).
17. Message Passing Forum, International Journal of Supercomputing Applications and High performance Computing, **8**(3-4), (1994).
18. Y.S. Wu, J. Liu, T. Xu, C. Haukwa, W. Zhang, H.H. Liu, and C.F. Ahlers, Report MDL-NBS-HS-000006, Lawrence Berkeley National Laboratory Berkeley, California; and Las Vegas, Nevada, CRWMS M&O, (2000).